スケーラブルラピッドプロトタイピングのための JIT-ORM
—DB アプリケーションの性能と開発効率を両立させる—

１．背景

オブジェクトリレーショナルマッピング(ORM)は高い開発効率やプラットフォーム間の移植性をもたらすため、データベースを利用するアプリケーション開発プロセスに欠かせない存在となっているが、その利便性の代償として N+1 SELECT 問題と呼ばれるパフォーマンス問題を常にはらんできた。ORM は基本的に遅延ロードでテーブルデータを読み出すため、入れ子ループ等の内部で大量のクエリが発行されてしまう。クエリ発行回数の増大はディスクランダムアクセス回数の増大を招き、読出し速度の劇的な低下や環境負荷増大の要因となる。

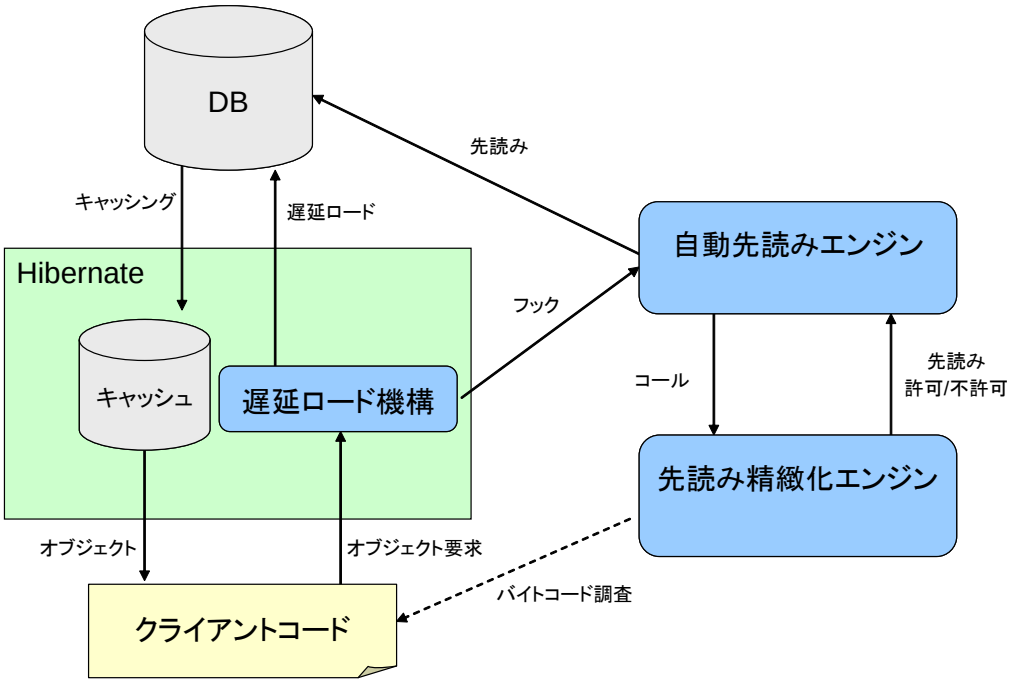
N+1 SELECT 問題への現状の対処法としては、ORM が大量発行する細切れの SQL を、同じ結果を一括して先読みするような少数のプリフェッチ SQL に手作業で変換する方法しかない。これにより DB は本来の性能を回復するが、この手法は開発効率の面から見て様々な欠点を持つ。SQL 記述の労力もさることながら、SQL のハードコーディングによってループの上下関係を完全に固定するために、クラス設計やループの構成を変える度に SQL も同時に修正せざるを得ないというメンテナンス性の著しい劣化を招く。また共通のサブループを切断して関数化できないため、SQL をループの組合せ数分書かなければならず、指数関数的な開発効率の低下を招く。

２．目的

上記のジレンマを解決するため、本プロジェクトでは現在手動で書かれているプリフェッチ SQL を内部的に自動生成するエンジンの開発を目標とした。また手法研究の域に留まらず、成果物を実用段階へと引き上げることも目標とした。

３．開発の内容

本プロジェクトでは、Java 上の既存 ORM である Hibernate を拡張することで目的を達成した。拡張内容は主に(1)自動先読みエンジン、(2)先読み精緻化エンジンからなる。システムの全体像を以下に示す。



(1) 自動先読みエンジン

従来プログラマが手作業で記述していたプリフェッチ SQL の内部的な自動生成・発行を行うためのモジュールである。

自動先読みエンジンは Hibernate 内部の遅延ロード機構からフックされ、発行されたクエリ情報を蓄積してゆく。また、蓄積済みのクエリ発行情報を用いてプリフェッチ SQL を構成可能であれば、プリフェッチ SQL を動的に生成し、それを DB に対し発行する。プリフェッチ SQL によって得られたオブジェクトは Hibernate 内部にキャッシュされ、以後 DB アクセスを伴うことなくクライアントコードに配付される。

(2) 先読み精緻化エンジン

自動先読みエンジンを採用した場合に起こりうる、データの過剰読み出しを未然に防ぐためのモジュールである。

先読み精緻化エンジンは、自動先読みエンジンが先読みを行う直前にコールされ、呼び出し側クライアントコードのバイトコードを分析する。バイトコードから部分制御フローグラフを構築し、過剰読出しとなるようなプログラム構造を含んでいるかどうかを検査する。もしそのようなコードが含まれていなければ先読みエンジンに対して許可メッセージを通知し、含まれていれば不許可メッセージを通知する。これにより先読みエンジンの動作を制御し、過剰読出しを防ぐ。

４．従来の技術（または機能）との相違

本成果物は、性能劣化を引き起こさずにソースコードから SQL 文を排除することを可能とし、開発効率とパフォーマンスの高度なレベルでの両立を可能とする。この特徴は従来の ORM には見られない新規性である。

本成果物を用いることで、プログラマは大量の SQL 文を記述/保守管理する手間を省くことができ、これは開発効率の大きな向上に繋がる。またプログラムの部品化が促進されるため、システム大規模化に伴う作業量の組み合わせ爆発を防ぐとともに、保守性・拡張性が高いプログラムを書くことが可能となる。さらに SQL の知識を直接的に必要としないため、開発に参加するエンジニアの技術レベル要請を押し下げることができ、大人数開発プロジェクトをスピーディなものとすることができる。

N+1 SELECT 問題によって、従来このような開発効率を得るためには処理速度の低下や環境負荷の増大が不可避であり、開発効率と性能は常にトレードオフの関係にあった。本成果物を用いることで、性能の劣化をほとんど引き起こさずに開発効率を向上させることができる。

本成果物は、ミッションクリティカルな業務アプリケーション等からの使用にも耐えうる信頼性と安定性を保持する。先読み精緻化エンジンによる正確な先読みデータ予測機構を有するため、システムにとって致命的となり得る潜在的な性能劣化要因を排除し、性能の下限を保証することができる。これによってプログラマは自身のプロジェクトに本成果物を安心して導入することができる。

以上のように、本成果物の導入は、多くのメリットがあるにも関わらず極めてデメリットが少ないということができる。また、機能の ON/OFF 設定をプログラム全体/部分において柔軟に切り替えることもできるため、一度試してみてもし気に入らなければすぐに使用をやめたり、必要部分のみ本成果物を用いたり、ということも柔軟にできる。

５．期待される効果

本成果物は DB アプリケーションの開発効率と性能を両立させることができる。これによって、手動最適化を行っていないラピッドタイピングアプリケーションはパフォーマンスの向上と環境負荷低減の効果が得られる。また、手動最適化済みのパフォーマンスアプリケーションは高い開発効率と保守性・拡張性を得る。

今回拡張を行った Hibernate は全世界で 290 万回以上もダウンロードされており、極めて普及度が高い。本成果物が Hibernate 本体にコミットされれば、その大多数が本成果物を利用することとなり、開発コスト面、リソースコスト面、環境面で非常に大きな効果が得られる。また仮にコミットが見送られた場合でも、半独立したモジュールとして提供することで世界中の Hibernate ユーザーに対し利便性と高パフォーマンスを提供できることになる。

さらに、今回は Hibernate/Java/RDB 上で実装したが、そこで用いた新手法自体はオブジェクト指向言語とデータベースの関係において汎用的なものであり、他言語/他 ORM/他ストレージに対しても適用できる可能性がある。それが実現できれば、本手法は一過性の技術としてではなく将来長い期間に渡って用いられることとなり、その総影響は極めて大きいものとなる。

６．普及（または活用）の見通し

現在、知人を通じて Hibernate コミット権を持つ方と連絡を取り、Hibernate 本体に本成果物をコミットするために折衝中である。これが実現した場合、Hibernate 使用アプリケーション 290 万個からの利用が見込める。またコミットが見送られた場合、半独立したモジュールとしてオープンソース化し提供することで、それに近い利用者を得ることを狙う。

また、上述の通り本プロジェクトで用いた手法は汎用性を持つため、本プロジェクトが終了し次第、Java/Hibernate/RDB 以外の新たな適用先を探してゆく。それと同時に今回得られた知見を論文化し、国際学会等で発表することも視野に入れる。

７．クリエータ名（所属）

長田　一登（東京大学大学院工学系研究科システム創成学専攻）
益子　遼介（東京大学工学部航空宇宙工学科）